Efficient Agent-Based Cluster Ensembles

Adrian Agogino
UCSC, NASA Ames Research Center
Mailstop 269-3
Moffett Field, CA 94035, USA
adrian@email.arc.nasa.gov

Kagan Tumer
NASA Ames Research Center
Mailstop 269-4
Moffett Field, CA 94035, USA
ktumer@mail.arc.nasa.gov

ABSTRACT

Numerous domains ranging from distributed data acquisition to knowledge reuse need to solve the cluster ensemble problem of combining multiple clusterings into a single unified clustering. Unfortunately current non-agent-based cluster combining methods do not work in a distributed environment, are not robust to corrupted clusterings and require centralized access to all original clusterings. Overcoming these issues will allow cluster ensembles to be used in fundamentally distributed and failure-prone domains such as data acquisition from satellite constellations, in addition to domains demanding confidentiality such as combining clusterings of user profiles. This paper proposes an efficient, distributed, agent-based clustering ensemble method that addresses these issues. In this approach each agent is assigned a small subset of the data and votes on which final cluster its data points should belong to. The final clustering is then evaluated by a global utility, computed in a distributed way. This clustering is also evaluated using an agent-specific utility that is shown to be easier for the agents to maximize. Results show that agents using the agent-specific utility can achieve better performance than traditional non-agent based methods and are effective even when up to 50% of the agents fail.

Categories and Subject Descriptors

I.2.11 [**Computing Methodologies**]: Artificial Intelligence—*Multiagent systems*

General Terms

Algorithms, Performance

Keywords

Clustering, Multiagent Systems, Reinforcement Learning

AAMAS'06, May 8-12, 2006, Hakodate, Japan.

1. INTRODUCTION

Clustering is a difficult problem in numerous fields including machine learning, multi-agent systems, data management and bioinformatics [14, 20, 11]. The central concept of clustering is that data points should be partitioned into separate clusters so that data points within a cluster are "close" to each other, while data points from different clusters are "far" from each other. A set of clusters with these properties is a clustering. To goal of the clustering problem is to find good clusterings. While the clustering problem is typically an NP-complete optimization problem, there are numerous heuristic clustering algorithms including k-means, expectation-maximization and Metis [13, 10, 2, 8]. Note that even though clustering is often used within multi-agent systems (e.g. to group similar agents), agents are not typically used to solve the clustering problem itself. This is as much due to the traditional view of a cluster as a passive label, rather than and active agent, as to the difficulty in decoupling the interactions among the agents representing the labels.

This paper focuses on using agents for an important subset of the clustering problem known as the *cluster ensemble* problem (Figure 1) [16]. In the cluster ensemble problem one needs to combine multiple clusterings, formed from different aspects of the same data set, into a single unified clustering. Cluster ensembles are particularly useful when all of the original data points are not available to create a clustering, but separate clusterings of the data still exist. This situation occurs:

1. When some of the data points come from proprietary sources, where the data owners are willing to reveal their clustering of the data, but not the data itself.
2. When the original data points are simply lost, or have been thrown away, but the much smaller summery cluster data is saved.
3. When all of the original data is available, but it may be too big to store in one site of computation. In this case, it may be desirable to make separate clusterings of different parts of the data and combine them later.
4. When we want to inject prior knowledge into learning systems or re-use information since new clusterings can be combined with older clusterings without needed to know all the information used in the previous clusterings or how they were derived.

Currently the best algorithms for cluster ensembles are graph-theoretic methods [15]. These methods treat the clus-

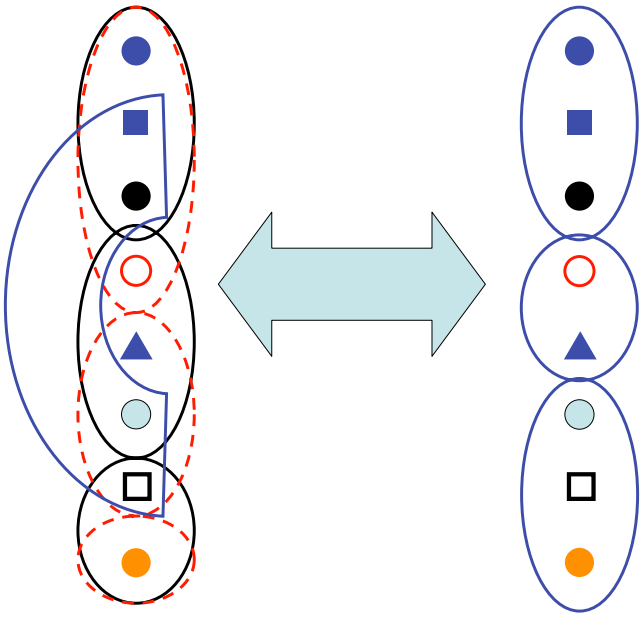

Figure 1: Cluster Ensembles. Shown on left are three different clusterings for eight data points. The goal of the cluster ensemble is to create a unified clustering (shown on right) that preserves the most information from the original clusterings.

ter ensemble problem as a global static problem, examining the entire data set and producing a single result. While these methods have shown to have good results on certain information theoretic measures, they tend to be inflexible [16]. The graph-based methods require the cluster combining to happen on a single node of computation. This requires all the data to be centrally available, creating a single point of failure. This centralization makes them inappropriate for domains that are inherently distributed, and have a potential for high component failure such as in space-based operations.

Instead of graph-based cluster ensemble methods, this paper proposes using learning agents to solve the cluster ensemble problem. In this approach each agent contributes to the final clustering and uses reinforcement learning to learn to maximize a utility function based on the clustering. Unlike graph-based methods, the multi-agent approach treats the cluster-ensember problem as a dynamic system, where agents continually attempt to create a unified clustering that maximizes a utility. This dynamic agent-based approach is more flexible and robust in the following ways:

- Agents are robust against noisy data sources.
- Agents naturally adapt to loss of data sources.
- Computation is distributed.
- There is no single point of failure, so performance gracefully degrades with the number of failures.
- Computation can be stopped at any time when performance is adequate.
- Data sources do not have to be revealed.

These advantages allow multi-agent cluster ensembles to be used more effectively in more domains than their non-agent-based counterparts.

Despite these advantages, multi-agent systems have not been used in the cluster ensemble problem due to coupling between the cluster assignments. The utility the agents attempt to maximize is highly non-linear, limiting the independence assumptions that can be made. This non-linearity prevents many multi-agent techniques from being applicable. In addition if an agent attempts to maximize the utility directly, it faces a difficult signal-to-noise problem in that it does not know the effect of its actions on the utility, among the effects of the actions of all the other agents. To address this issue, this paper shows how agents can instead use agent-specific utilities that are much easier to maximize, yet are still aligned with the original utility.

In Section 2, this paper gives a formal description of the cluster ensemble problem as well as a brief overview of a graph-based and simple greedy solutions to the cluster ensemble problem. In Section 3 this paper presents an agent-based approach. Finally in Section 4 the paper shows results for the performance of agent-based cluster ensemble methods, and how they are robust against agent failure.

2. CLUSTER ENSEMBLE PROBLEM

In clustering problems a set of data points are grouped into clusters, so that data points in the same cluster are "closer" to each other than data points from different clusters. The closeness between data points is typically measured with a domain dependent distance metric. This metric is influenced by the type of data, the goal of the clustering and even the "scale" of the clustering depending on the amount of clusters desired [17, 16, 4]. Given a distance metric, a "soft-clustering" can be performed where data points belong to different clusters with varying probability, or hard clustering where each data point is part of only one cluster [6]. This paper will focus only on hard-clustering. This type of clustering problem is equivalent to creating a non-overlapping partition of the data. In this paper we refer to a particular partition as a *clustering*. In general there can be many different clustering for the same data, depending on the clustering algorithm used [12, 5, 8]. In addition there can be sets of related clusterings that are formed with overlapping subsets of the data [16]. The goal of the cluster ensemble problem is to create a single clustering that best characterizes a set of clustering, without using the original data points used to generate the clusterings.

To formalize the clustering ensemble problem, we need a way to compare two clusterings. In this paper, this is done through an information theoretic measure previously used in the clustering community called *normalized mutual information* (NMI) [16]. This measure is based on the sizes of the clusters within the clusterings (a formal justification for this is presented in [15]). To compute this measure we fist define a clustering X, where each cluster $x_i \in X$ is a set of data points. We now define the mutual information between clusterings X and Y as:

$$I(X,Y) = \sum_{x \in X} \sum_{y \in Y} \frac{|x \cap y|}{n} log_2 \left(\frac{n|x \cap y|}{|x||y|} \right) \quad (1)$$

where there are n data points and $|\cdot|$ and $\cap$ are the set cardinality and intersection operators respectively. The in-

tersection operation $x \cap y$ returns the data points that are common to both clusters x and y. Next we define the entropy of a clustering, X, as:

$$H(X) = -\sum_{x \in X} \frac{|x|}{n} log_2 \left(\frac{|x|}{n} \right) \quad (2)$$

Now we can define the normalized mutual information (NMI) between two clusters, X and Y as:

$$\mathrm{NMI}(X, Y) = \frac{I(X, Y)}{\sqrt{H(X)H(Y)}} \quad (3)$$

which has the desirable property of being bounded by [0, 1] and having $\mathrm{NMI}(X, X) = 1$.

The normalized mutual information between two clusters measures how much information they have in common. If two clusterings are similar they will have an NMI close to one. If two clusterings are completely different then they have an NMI of zero. In this case knowing one clustering would not help one predict the clusters within the second clustering. If we had a "true" clustering of the data we would want our ensemble clustering to have as high as possible NMI with respect to this true clustering. However, typically all we have to compare our ensemble clustering with is the original clusterings used to create the ensemble. In this case our goal is to create an ensemble clustering that shares as much information as possible with the clusterings it was created from. This can be measured by averaging the normalized between the original clusterings and the ensemble. Formally we define the ANMI between a clustering X and a set of clusterings $\overline{Y}$ as:

$$\mathrm{ANMI}(\overline{Y}, X) = \frac{1}{|\overline{Y}|} \sum_{Y \in \overline{Y}} \mathrm{NMI}(X, Y) \ . \quad (4)$$

When an ensemble clustering X has more information in common with the original clusterings $\overline{Y}$, $\mathrm{ANMI}(\overline{Y}, X)$ will have a higher value. Therefore the goal of the agents is to find the clustering, X^*, that maximizes the ANMI between X^* and the set of available clusterings $\overline{Y}$.

2.1 Graph-based Approaches

The graph-based approaches to the cluster ensemble problem involve representing the data points as nodes in a hypergraph, and the clusters as undirected hyperedges of the hypergraph. This paper will compare cluster ensemble performance with three of these algorithms, which are described in detail in [16]. The first algorithm called CSPA (Cluster-based Similarity Partitioning Algorithm) provides moderate performance, but requires significant computation. CSPA works by creating a similarity measure between data points. Within a clustering, all data points in the same cluster have a similarity of 1, and data points from different clusters have a similarity of 0. If the similarities between data points are averaged over different clusterings, a new single clustering can be made with a similarity based clustering algorithm, such as METIS [8].

The second clustering algorithm is called HGPA (Hyper-Graph Partitioning Algorithm) and is simpler to compute than CSPA. This method creates the final combined clustering for the ensemble by using HMETIS [9] to perform hypergraph partitioning on the hypergraph that represents the clusterings. The objective of this method is to create clusters that break the least number of hyperedges. The final method, called MCLA (Meta-CLustering Algorithm), provides better performance than HPGA and retains its low computational complexity. This algorithm works by collapsing a group of related hyperedges into a single hyperedge. This can be seen as clustering clusters.

2.2 Simple Greedy Optimization

Simple greedy optimization approaches have been applied to the cluster ensemble problem in [16]. In this method, one starts with a single representative clustering, X, which is usually the clustering that has the highest ANMI with respect to all the other clusterings:

$$X = argmax_{X' \in \overline{Y}} \mathrm{ANMI}(\overline{Y} \backslash X', X') \quad (5)$$

where $\overline{Y}$ is the set of clusterings, and $\backslash$ is the set difference operator. For each data point, a new clustering is created from the previous clustering by moving the data point to a new cluster at random. If this new clustering has a higher ANMI than the previous one, then it is preserved, otherwise it is thrown away. The algorithm is repeated for each data point. When it has gone through all the data points, the algorithm stops if all of the new clusterings where thrown away. Otherwise it repeats through all the data points. This algorithm can be seen as a form of serial simulated annealing with zero exploration or as a Stackelberg game [7]. Unlike the graph-based methods, this algorithm is dynamic as it continually tries to maximize utility. However, this algorithm has a number of difficulties. Since the exploration is zero, the system will only reach a local maximum. Also since for each loop, the ANMI has to be computed for every data point, the computationally burdensome when there are many data points.

3. AGENT-BASED APPROACH

Similar to the greedy optimization approach, the agent-based approach treats the cluster ensemble problem as a dynamic optimization problem, with the agents striving to create a final combined clustering with the highest utility. However, the agent-based approach requires significantly less computation and is naturally distributed. In this approach agents are assigned a set of data points and vote on which cluster in the final clustering the data points should belong two. Through reinforcement learning, the agents learn how to vote in a way that leads to the best final clustering.

3.1 Assigning Agents and Voting

In this paper, the cluster ensemble problem is approached by assigning one or more agents to each cluster in the original set of clusterings. If m agents are assigned to each cluster, then the multi-agent sytem for r clusterings with k clusters each would have mrk agents. The action of an agent is to vote on the cluster in the combined clustering, to which the data points it is responsible should belong. A data point will then belong to final cluster that received the most votes from agents that were responsible for the cluster containing the data point. This process is shown in Figure 2. The global utility of the system is the ANMI of the combined clustering with respect to the original clusterings. Agents can then try to maximize the ANMI, by using their private utilities to help them choose the best actions. In this example, the top two points are in the same clustering in two of the three clusterings (agents 1 and 4). Similarly,

the second and third data points are in the same clustering in two of three clusterings (agents 1 and 8). In the final clustering, all three data points belong to the same cluster, as this clustering minimizes the mutual information between the original clusterings.

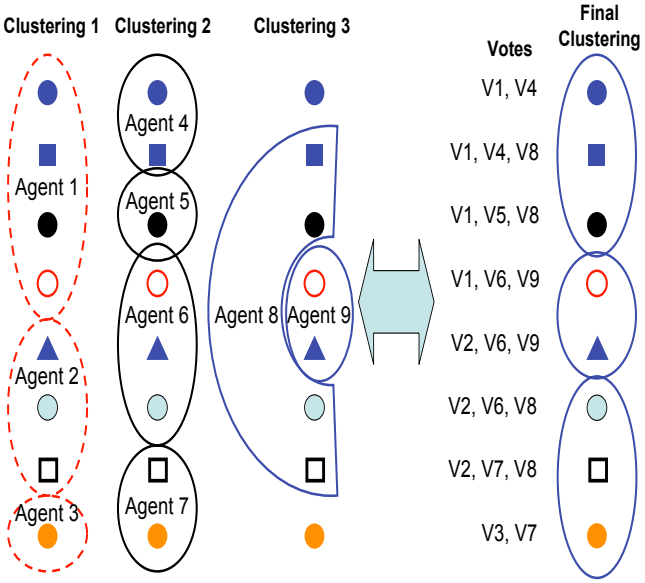

Figure 2: Agent-Based Cluster Ensembles. An agent is assigned to a cluster in each clustering. Agents then vote on the final cluster to which the data points within their current cluster belong.

Formally the votes by the agents for a combined clustering Z can be represented by a two dimensional array of sets indexed by agents and clusters. If agent i chooses cluster $z \in Z$, then $V_{i,z}$ equals the set of data points in agent i's assigned cluster. If agent i did not choose z then $V_{i,z}$ equals the empty set. The number of votes for data point p to be put in cluster z can be formulated as follows:

$$N_{p,z} = \sum_i I_{p \in V_{i,z}} \tag{6}$$

where I is an indicator function. The assignments of data points to a cluster in the combined clustering can now be expressed as:

$$z = \{p | N_{p,z} = max_{z'} N_{p,z'}\} \tag{7}$$

3.2 Global Utility

The global utility that the agents ultimately try to maximize is the ANMI between Z and the initial set of clusterings $\overline{Y}$:

$$G(\overline{Y}, V) = \text{ANMI}(\overline{Y}, Z(V)) \; . \tag{8}$$

Note that the global utility can be computed in a distributed way when the votes V are broadcasts to all agents. Each agent can compute the final clustering $Z(V)$ and compute its own normalized mutual information NMI(Y, Z(V)). The NMIs can then be broadcasts and all the agents can compute ANMI and therefore G on their own. While this process requires a fair amount of communication, it is usually considerably less than broadcasting the entire data set, depending on the size of the clusters. While an agent can compute the global utility, it is difficult for an agent to maximize the global utility directly since this utility is a function of the actions of all of the other agents. If an agent takes an action and the global utility increases, the agent does not know if the increase is due to its action or the actions of other agents. As a consequence, when there are many agents, it will take a large number of learning steps for an agent to discern the effects of its actions from the actions of all the other agents.

Note that while the computation of G ultimately uses information derived from all of the data points, these data points are never assembled into a central data set. Instead the computation of G involves the agents broadcasting what they believe the final clustering should be, not what their current clusterings are. In this distributed framework the original clusterings do not have to be revealed. This decentralized computation allows the agent-based approach to be used in domains where the owners of individual data sources do not want their data set to be made fully available to the other data owners.

3.3 Agent-Specific Utility

Instead of maximizing the global utility directly, an agent can learn more quickly if it maximizes an agent-specific "private" utility that is more influenced by its own actions than the actions of all the other agents. We call this relative influence an agent has on its utility, the "learnability" of the utility [18]. Learnability for a private utility g_i for agent i can be informally defined as ratio of agent i's influence on g_i to all other agents' influence on g_i. Generally if a utility is more learnable for an agent, the faster an agent will be able to maximize it. However, having all the agents being able to maximize their private utilities is not useful if this does lead to the maximization of the global utility. For the maximization of private utilities to lead to the maximization of the global utilities, each private utility should be "factored." A private utility is factored when any action an agent takes to increase its private utility also increases the global utility [18, 21].

A utility that has been shown to have high learnability while being factored is the difference utility, defined as:

$$D_i(z) = G(z) - G(z_{-i}); \tag{9}$$

where z are the actions of all the agents and z_{-i} are the actions of all the agents other than agent i. The second term of the difference utility is a counterfactual, computing the value of the system without agent i. Subtracting this counterfactual from the global utility therefore computes an agent's contribution to the global utility. This utility is factored since the second term is not a function of the agent's actions, therefore it can only influence the utility by changing the value of the first term, the global utility. Furthermore, this utility usually has far better learnability than does $G(z)$ because the second term of D_i removes a lot of the effect of other agents (i.e., noise) from agent i's utility [18]. This utility has proven effective in many multi-agent system domains including network routing, rover control, job scheduling and congestion games [1, 19, 18].

Using the global utility defined in 8 we can define the difference utility for the cluster ensemble problem as:

$$\text{D}_\text{i}(\overline{Y}, V) = \text{ANMI}(\overline{Y}, Z(V)) - \text{ANMI}(\overline{Y}, Z(V')) \tag{10}$$

where V' is the same as V, except that all of the elements

of V' that are indexed by i are equal to the empty set. Since this utility is factored, agents maximizing $D_i(\overline{Y}, V)$ will tend to maximize $G(\overline{Y}, V)$, though they can learn more quickly since each agent has more influence over its own difference utility than over the global utility. Note that when an agent's vote does not influence the final clustering Z(V') the difference utility does not have to be computed since its value is zero. In large systems, this case will occur often. If an agent's vote is a deciding vote, then all the agents will have to compute and broadcasts their NMIs for the counterfactual $Z(V')$. In the worst case when all the votes are deciding votes, the amount of computation grows linearly with the number of agents. However, this system is still distributed and an agent can simply not average in NMIs from other agents that fail to respond. This distribution is critical to robustness as agents can still try to maximize their utilities even when some of the agents are not working properly.

4. RESULTS

The performance of the multi-agent approach to cluster ensembles relative to other methods was tested with three data sets. The first data set was artificial, while the other two came from real-world problems. These data sets showed different aspects of the cluster ensemble problem. In the first data set, the "true clustering" was known since it was artificially generated. The second data set involved clustering images of hand-written images, where the true number of clusters was known, but the correct clustering was subject to interpretation. The final data set involved groupings in the Yahoo! web site, where both the correct number of clusters and the correct clustering was subject to interpretation. The experiments showed that the multi-agent approach often achieved superior performance to the best existing centralized cluster ensemble algorithms, while being able to recover from a number of types of agent failures.

All agents learned with a simple single-time-step reinforcement learner. This learner was equivalent to an Q-learner with infinite reward discounting ($\gamma = 0$) so that only the immediate reward was processed. At every time step, each agent would choose one of k clusters based on its estimates of the utility for that choice. These estimates were stored in a utility table of size k. This process was done in an ϵ-greedy fashion (with $\epsilon = 0.005$) where the highest valued cluster would be chosen with probability $1 - \epsilon$ and a random cluster would be chosen with probability ϵ. After all the agents chose their cluster, each agent would compute their private utility and use it as the reward to update their reinforcement learner.

4.1 Artificial Data

In the first experiment, four hundred data points where randomly placed into ten clusters to form an initial clustering. From this initial clustering, eight variants were created. Each variant was made by first duplicating the original clustering and then adding random noise by moving a random subset of the data points to new clusters (for each data point in the subset, the cluster it was moved to was chosen independently over a uniform distribution over all the clusters). The amount of random noise was specified by B, the fraction of all the data points contained in the subset. These noisy variant clusterings were used as the ensemble clusterings and experiments were performed with B ranging from 0.0 to 1.0. The performance of the algorithms were determined by the NMI between the clustering produced by the algorithm and the initial clustering. Since this is an artificial data set we have "ground truth" about what the true clusterings should be. A better cluster ensemble algorithm should produce a clustering as close to possible to the initial clustering in which the eight ensemble clusterings were derived. Figure 3 shows the results with $B = 0.4$ for agents using D_i or G as their utility along with results obtained from the graph-based methods and the simple greedy method.

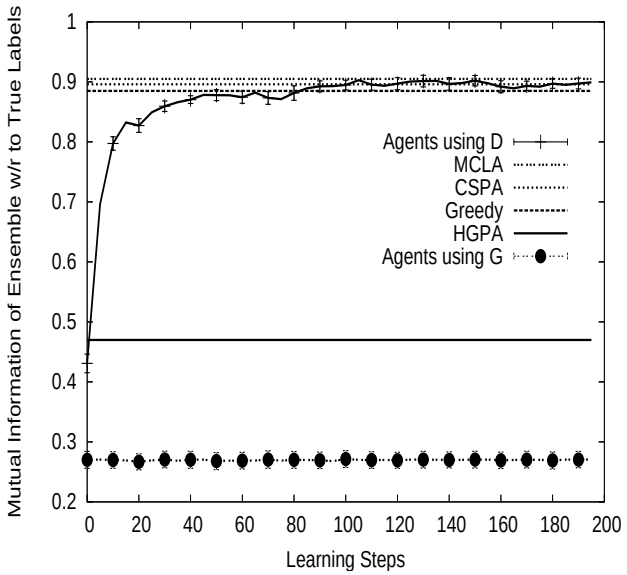

Figure 3: Results of Ensemble Methods with Artificial Data. Eight clusterings are created by adding 40% noise to the original cluster labels. The ensemble tries to recover the original labels from the eight clusterings. The decentralized algorithm using the difference utility performs as well as the best graph-based algorithm, which requires centralization.

The results show that the agent-based method using the difference utility performed as well as the best graph-based method, MCLA and considerable better than the lowest performing graph-based method, HGPA. In noisy domains HGPA had particular poor performance as compared to MCLA because MCLA could reduce noise by collapsing similar edges. In addition to performing as well as the best graph-based methods, the agent-based method using the difference utility performed about the same as the simple greedy method. However this performance was achieved with much less computation. The greedy method was reported to take an hour on a 1Ghz PC [16]. The agent-based method took only 45 seconds on a similar computer. In contrast to agents using the difference utility, agents using the global utility performed very poorly. This poor performance can be attribute to the large number of agents in this problem (eighty). When an agent choose a cluster and the value of the global utility changed, the agent did not know whether the change was caused by its action or the action of one of the seventy nine other agents. This very low performance of the global utility is likely the reason why agent-based cluster ensembles have not been pursued until now.

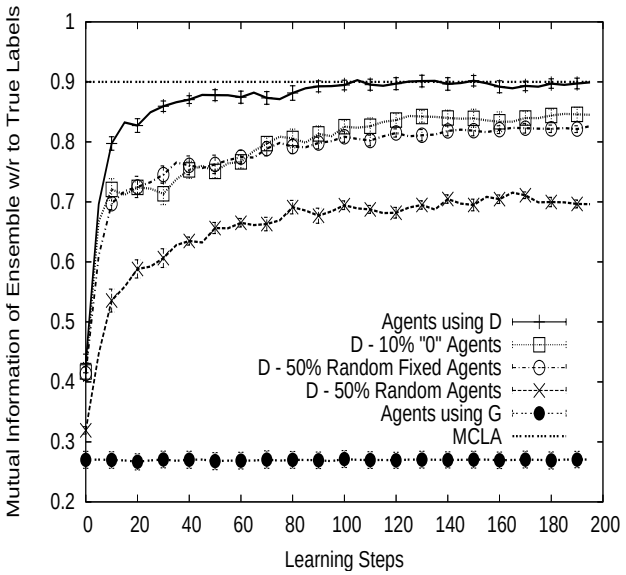

Figure 4: Results of Ensemble Methods with Artificial Data with Faulty Agents. Systems with agents using D generally perform well when some of the agents fail. Top curve shows performance when all agents are working properly. Second curve shows performance when 10% of the agents always choose cluster 0 as their action. Third curve shows performance when 50% of the agents always choose the same random cluster as their action. Fourth curve shows performance when 50% of the agents always choose a different random action at every time step.

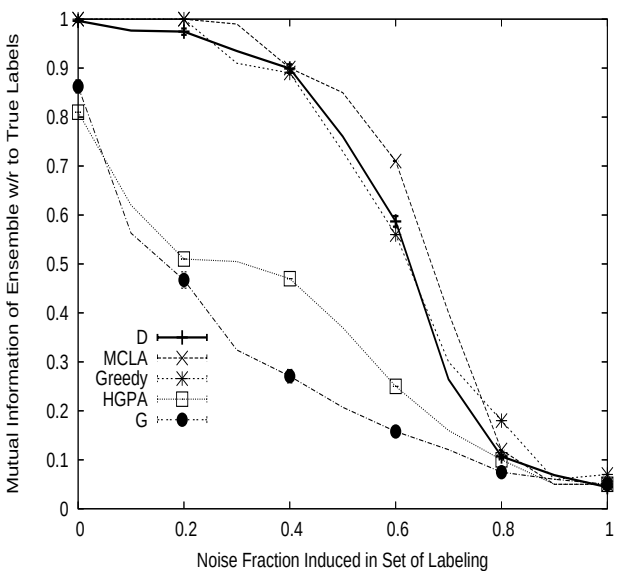

Figure 5: Results of Ensemble Methods on Artificial Data. Eight clusterings are created by adding variable amounts of noise to the original cluster labels. The ensemble tries to recover the original labels from the eight clusterings. Relative performance of methods is consistent across noise levels (CSPA performance is almost identical to greedy method, but not shown to reduce clutter).

In addition to testing the agents under normal conditions, we tested the case where some of the agents fail. In this test three agents were used for every cluster to provide redundancy. This redundancy was tested under three failure scenarios. In the first scenario 50% of the agents were faulty. These faulty agents (called "random agents" in the figures) chose a random cluster at every time step. In the second scenario 50% percent of the agents (called "fixed random agents") chose a random cluster at the beginning of a trial and kept making the same choice throughout the trail. In the final scenario 10% of the agents always chose the first cluster (these are called "0" agents). The results shown in Figure 4, reveal that agent-based cluster ensembles can perform well in a number of adverse conditions. Even when half of the agents were faulty, the system could still perform well. Note that the agent-based system in the first scenario performed worse than the one in the second scenario, even though the same number of agents were faulty. This can be explained by the adaptivity of the agents. In the second scenario the faulty agents always took the same wrong move, so the working agents could adapt to overcome the adversity. In the first scenario, the faulty agents simply added random noise to the system, which could not be easily adapted to. Despite the inherent failure tolerance of the multi-agent system, there was still a possibility of catastrophic failure if too many of the agents failed in the same way. While the system was able to recover when 10% of the agents choose the same wrong cluster, the performance drops to zero if 50% percent of the agents choose the same wrong cluster (not shown in the graphs). This failure is caused by the voting scheme, since when 50% percent of the agents vote for the same cluster, that cluster will always win. In this case, the final clustering will always have just one cluster.

The previous experiments were done with the noise parameter, B, used to generate the noisy clusterings from the initial clustering, set to 0.4. To show that the relative performance of all the methods was not dependent on this choice of noise parameter, we tested the performance of the methods on a wide range of values shown in Figure 5. The results show that for a wide range of noise parameters that the agent-based methods perform well, and the relative performances of the other methods is about the same.

4.2 Pendig Data Set

The Pendig data set contained one thousand data points, with each data point having sixteen features. Each data point represented a processed handwritten digit. The data points were clustered into ten clusters, with each cluster representing a digit from '0' to '9.' While this data set is known to have precisely ten clusters, the true clustering is subjective since some of the images could be ambiguous. From the original Pendig dataset we produced ten different clusterings (by coincidence, each with ten clusters) by partitioning the data into ten data sets. Each point in the new data sets contained four features sampled from the original sixteen. Ten clusterings were then produced by clustering each of the new data sets with METIS. Using these clusterings for the ensemble, agents using the difference utility were able to perform better than the best centralized graph-based algorithm, MCLA (see Figure 6). Again the results show that

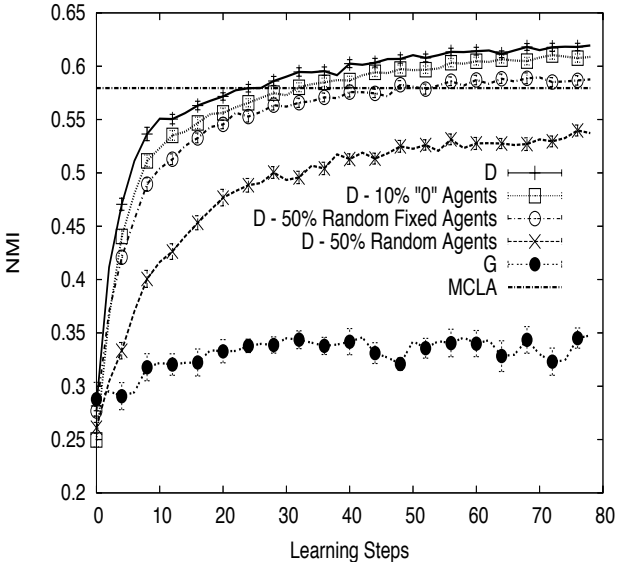

Figure 6: Results of Ensemble Methods for the Pendig Data Set. Agents using difference utility perform better than the best graph-based algorithm. Even when up to 50% of the agents fail, the system performs well.

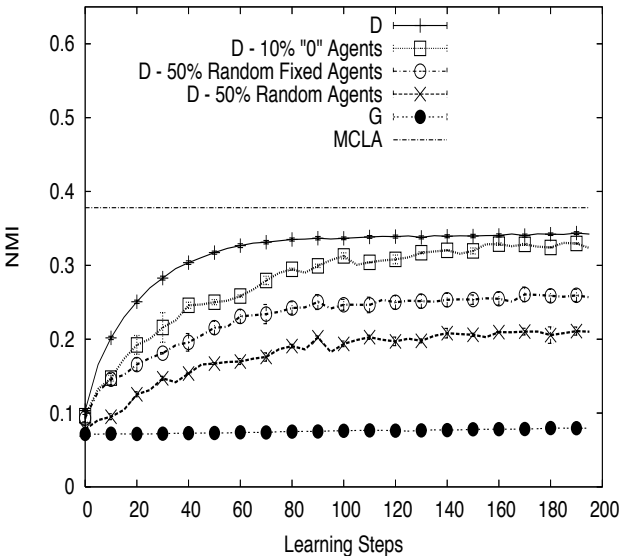

Figure 7: Results of Ensemble Methods for the Yahoo Data Set. Agents using difference utility slightly under-perform the best graph-based algorithm, but still perform adequately when up to 50% of the agents fail.

agent-based cluster ensembles can perform well in a number of adverse conditions. Even when half of the agents were faulty, the system could still out-perform the graph-based methods. Only in the case when half the agents choose random actions at every time step, did the performance drop below that of the MCLA. Though even in this case the agents using the difference utility with 50% failure were able to perform better than agents using the global utility when there are no failures.

4.3 Yahoo Data Set

In this experiment a processed version of the Yahoo! data set was used. This data set has also been used in [17] and [3]. The data set contained 2340 data points, with each data point having 2903 features. Each data point represented a document and the features were a pruned set of word frequencies contained in the document. The data set was clustered into twenty clusters, based on which Yahoo! news category they were originally placed in. Note that both the correct number clusters and the true clusterings were subjective for this data sets. More or less news categories could have been used, and many documents could have been placed in different categories.

In the experiment we created twenty clusterings (coincidentally, each clustering having twenty clusters) by splitting the full data set into twenty separate data sets. Each new data set contained all of the 2340 data points, but each data point only had a 128 element subset of the original features. Each new data set was then clustered using METIS, forming twenty clusterings. These new clusterings were then used for the ensemble in the experiment. The output of the ensemble algorithm was then compared to the original Yahoo! clustering using NMI. Figure 7 shows the results between an agent-based method using D_i as its utility, an agent-based method using G as its utility and the best performing graph-based method, MCLA.

The results show that agents using the global utility were unable to produce an adequate clustering. This is not surprising since there were 400 agents in the system. In this case, the learnabity of the global utility was very poor since an agent cannot easily see the contribution of its action on the global utility, amongst the 399 actions of the other agents. In contrast the agents using D_i were able to do much better. While these agents were not able to do as well as the MCLA algorithm [1], they did exhibit a high level of fault tolerance. When 10% of the agents were not working properly, the system still managed to achieve nearly the same performance as the fully operational system. Even when 50% of the agents were taking random actions, agents using the difference utility were still able to learn to compensate, and achieve much better performance than the system using the global utility.

5. DISCUSSION

Using the difference utility, the results show that agent-based cluster ensembles can perform comparably to the best existing cluster ensemble methods. In addition they are able to achieve this level of performance with relatively little computation as compared to simple greedy optimization methods. Also agent-based method also allow the cluster ensemble problem to be treated as a dynamic optimization problem, significantly increasing the robustness of the system. The agent-based cluster ensembles exhibit a high level

[1]Note that we could not generate the exact clusterings used in the papers [16, 15], due to the randomness in the Metis algorithm. Our results for MCLA and the agent methods were based on a set of clusterings that turned out to be slightly harder, therefore our results cannot be directly compared to the results in this paper.

of fault tolerance and are able to retain high performance even when 50% of the agents fail. This property allows them to be used in domains that have a high component failure rate, such as in space-based systems or domains with unreliable communication. In addition the agent-based cluster ensembles have the flexibility to be stopped early in domains where a fixed level of performance is needed. This ability can save computational costs, even when the difficulty of the ensemble problem is not known ahead of time.

The implementation presented in this paper assigned an agent to a single cluster and the agent vote involved all the data points in that cluster. However, this way of mapping agents to data points is not necessary to the framework. Depending on the performance requirements one agent could be assigned several clusters or multiple agent could be assigned to different parts of the same cluster. Having alternative agent to cluster mapping may be beneficial if the owners of the clustering sources do not want to reveal their exact clusterings, but still want to collaborate with other data owners to create unified clusterings.

6. ACKNOWLEDGMENTS

We thank Joydeep Ghosh and Alexander Strehl for their help in this project.

7. REFERENCES

[1] A. Agogino and K. Tumer. Efficient evaluation functions for multi-rover systems. In *The Genetic and Evolutionary Computation Conference*, pages 1–12, Seatle, WA, June 2004.

[2] Christopher M. Bishop. *Neural networks for pattern recognition*. Oxford University Press, 1996.

[3] D. Boley, M. Gini, R. Gross, S. Han, K. Hastings, G. Kary pis, V. Kumar, B. Mobasher, and J. Moore. Partitioning-based clustering for web document categorization. *Decision Support Systems*, 27(3):329–341, 1999.

[4] S. V. Chakravarthy and J. Ghosh. Scale-based clustering using the radial basis function network. *IEEE Trans. on Neural Networks*, pages 1250–61, September 1996.

[5] I. S. Dhillon and D. S. Modha. Concept decompositions for large sparse text data using clustering. *Machine Learning*, 42(1):143–175, 2001.

[6] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification (2nd ed)*. John Wiley & Sons, New York, NY, 2000.

[7] D. Fudenberg and J. Tirole. *Game Theory*. MIT Press, Cambridge, MA, 1991.

[8] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1998.

[9] George Karypis, Rajat Aggarwal, Vipin Kumar, and Shashi Shekhar. Multilevel hypergraph partitioning: Applications in vlsi domain. In *Proceedings of the Design and Automation Conference*, 1997.

[10] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *The Bell System Technical Journal*, 42(2):291–307, 1970.

[11] J. Kim and T. Warnow. Tutorial on phylogenetic tree estimation. In *"Intelligent Systems for Molecular Biology"*, Heidelberg, Germany, 1999.

[12] T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43:59–69, 1982.

[13] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, 1967.

[14] T. M. Mitchell. *Machine Learning*. WCB/McGraw-Hill, Boston, MA, 1997.

[15] Alexander Strehl. *Relationship-based Clustering and Cluster Ensembles for High-dimensional Data Mining*. PhD thesis, The University of Texas at Austin, May 2002.

[16] Alexander Strehl and Joydeep Ghosh. Cluster ensembles – a knowledge reuse framework for combining partitionings. In *Proceedings of AAAI 2002, Edmonton, Canada*, pages 93–98. AAAI, July 2002.

[17] Alexander Strehl, Joydeep Ghosh, and Raymond J. Mooney. Impact of similarity measures on web-page clustering. In *Proc. AAAI Workshop on AI for Web Search (AAAI 2000), Austin*, pages 58–64. AAAI/MIT Press, July 2000.

[18] K. Tumer and D. Wolpert, editors. *Collectives and the Design of Complex Systems*. Springer, New York, 2004.

[19] K. Tumer and D. H. Wolpert. Collective intelligence and Braess' paradox. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, pages 104–109, Austin, TX, 2000.

[20] Ian Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, San Francisco, 2005.

[21] D. H. Wolpert and K. Tumer. Optimal payoff functions for members of collectives. *Advances in Complex Systems*, 4(2/3):265–279, 2001.